Supervised Capstone

Google Play Store App: Regression & Classification Analysis

Arun Nair

Thinkful Data Science Immersive

Overview : Google Play Store Dataset.

Features:

- Apps, category, ratings, reviews, price etc.

Challenges:

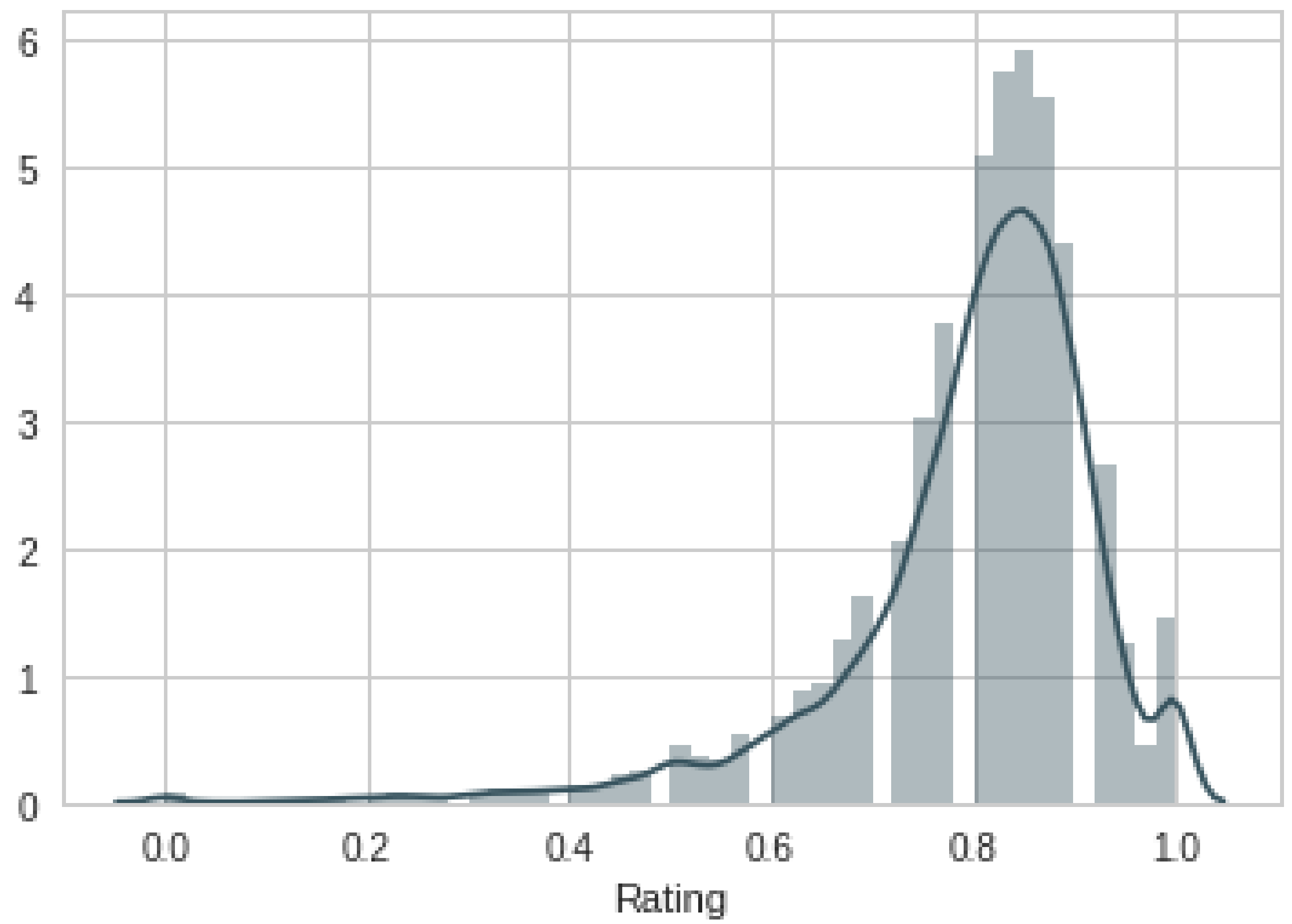
- Web Scraping tough compared to other Datasets.
- Cleaning dirty data.
- Converting data to numeric and finding tangible correlation.

Goals (Research Topic):

- To accurately predict ratings against Play Store parameters.
- Classify apps based on ratings vs. categorical parameters.

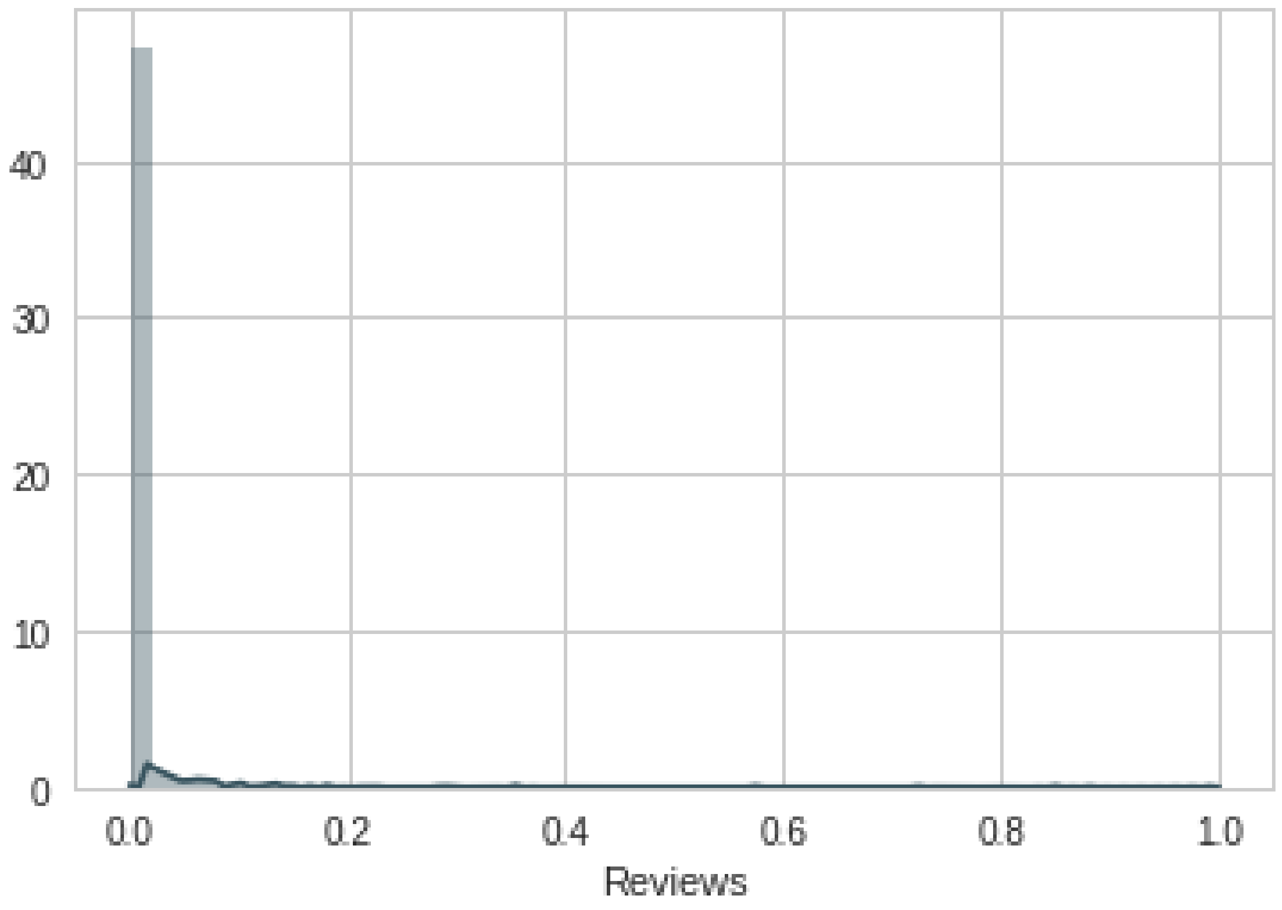
Exploratory Data Analysis

- **Ratings**
  - Long-tailed Distribution.
  - Rating Score: 1-5.
- **Central Tendency Measures:**
  - Mean: 4.19
  - Median: 4.3
  - Mode: 4.4



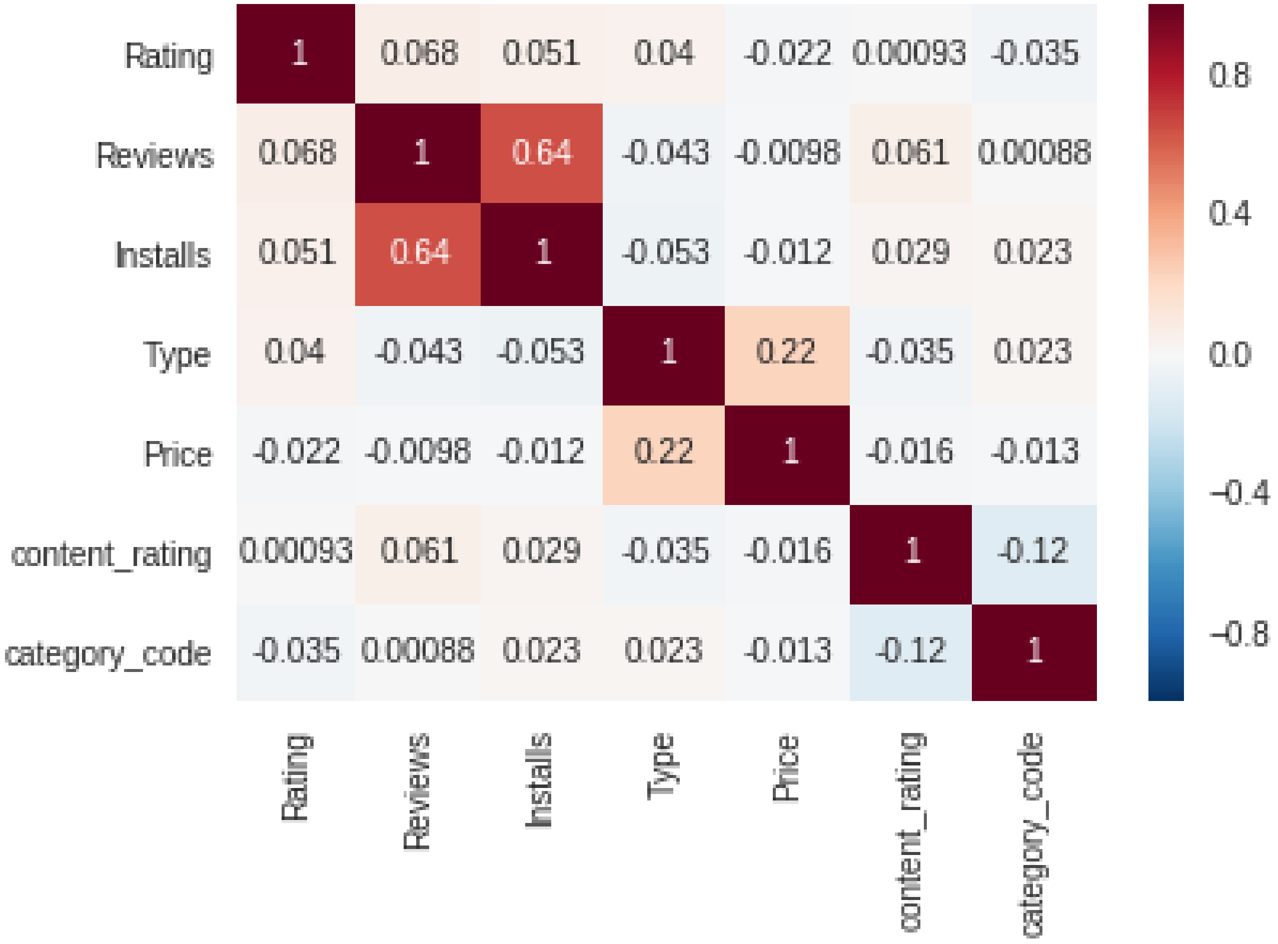
Exploratory Data Analysis

- **Reviews**
  - Long-tailed Distribution.
  - Reviews: 1-5.
- **Central Tendency Measures:**
  - Mean: 4.44152+05
  - Median: 5930.5
  - Mode: 2



Correlations

- Highest correlation between Installs and Reviews.
- Negative correlation between Price and most variables.



Feature Engineering

- **Feature Selection:** Reviews, Installs, Type, Price, Category, Content Rating.

- Pre-modeling for Classification and Regression:
    - Classification: Converting continuous x parameters to categorical data type.
    - Regression: To apply logistic regression, we binarize ratings as being above or below the median.

Modeling - Regression

- features = ['Reviews’, 'Installs’, 'Type’, 'Price’, 'category_code','content_rating']
- X = google_scaled[features]
- y = google_scaled[['Rating’]]

- **<u>Training the Model:</u>**
- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

Linear Regression

- Training Evaluation:
  - Intercept: 0.80417688
  - Coefficients:
    - 'Reviews’: 0.17311278
    - 'Installs’: 0.02341905
    - 'Type’: 0.02615527
    - 'Price’: -0.09135575
    - 'category_code’: -0.01950176
    - 'content_rating’: 0.00222041

- Testing Evaluation:
  - Mean Squared Error = 0.08931495536371267
  - Mean Absolute Error = 0.016007898366191792
  - Root Mean Squared = 0.12652232358833673

Ridge Regression

- The r^2 value is = 0.008564418905254056
- Coefficients:
  - 'Reviews’: 0.1618027
  - 'Installs’: 0.02650054
  - 'Type’: 0.02595659
  - 'Price’: -0.08714568
  - 'category_code’: -0.01947737
  - 'content_rating’: 0.00234455

Polynomial Regression

Now we transform the original input data to add polynomial features up to degree 2 (quadratic)

Addition of many polynomial features often leads to overfitting, so we often use polynomial features in combination with regression that has a regularization penalty, like ridge regression.

```
(poly deg 2 + ridge) linear model coeff (w):
[[ 0.00000000e+00  2.78950506e-01  2.16420903e-01  6.63800203e-03
  -3.40277212e-02  5.28231530e-02 -1.86844155e-02 -1.56101414e-01
  -1.54042240e-01  3.61885309e-03  4.40025691e-05  1.12582907e-01
   2.39671586e-02 -2.62963228e-01  2.17005644e-03  1.88825852e-05
   5.51066201e-02 -2.02420052e-02  6.63800203e-03 -3.40277212e-02
   1.85762907e-02  6.29007300e-02 -1.33086994e-03 -5.11846340e-02
   1.17069304e-02 -7.25191588e-02  1.25360705e-01 -1.13356854e-01]]
(poly deg 2 + ridge) linear model intercept (b): [0.79159375]
(poly deg 2 + ridge) R-squared score (training): 0.02244357263766372
(poly deg 2 + ridge) R-squared score (test): 0.024319713337627835
```

Modeling - Classification

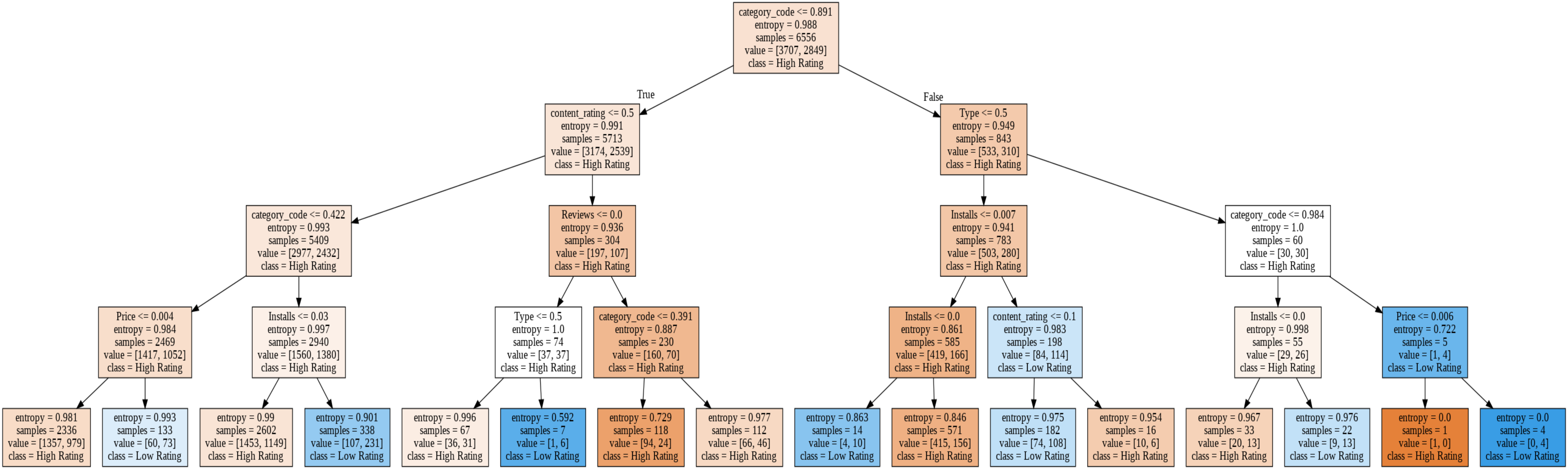
```
y_train_binary = (y_train > y_train.median()).astype(np.int)
y_train_binary.head(3)
```

	Rating
558	1
1891	1
5626	0

Logistic Regression

	precision	recall	f1-score	support
0	0.58	0.94	0.72	1595
1	0.57	0.10	0.16	1215
micro avg	0.58	0.58	0.58	2810
macro avg	0.57	0.52	0.44	2810
weighted avg	0.58	0.58	0.48	2810

Decision Tree



Random Forest

	precision	recall	f1-score	support
0	0.72	0.74	0.73	1595
1	0.65	0.62	0.63	1215
micro avg	0.69	0.69	0.69	2810
macro avg	0.68	0.68	0.68	2810
weighted avg	0.69	0.69	0.69	2810

Conclusion

- R^2 is low in regression analyses.
- Polynomial features improve the performance, but overall score remains low.
- Random Forest had best result with 68% average accuracy.
- Continue to work on variables and manipulate feature parameters.